

Titel ausstehend

## STUDIENARBEIT

für die Prüfung zum  
Bachelor of Engineering  
des Studienganges Informationstechnik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

**Tim Saupp**

Abgabedatum 18.09.2017

Bearbeitungszeitraum  
Matrikelnummer  
Kurs  
Gutachter der Studienakademie

03.07.2017-15.09.2017  
2742603  
TINF15B3  
Prof. Dr. Lausen

## Erklärung

Ich versichere hiermit, dass ich meine Studienarbeit mit dem Titel: Titel ausstehend selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

---

Ort   Datum

---

Unterschrift

## **Sperrvermerk**

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anders lautende Genehmigung der Ausbildungsstätte vorliegt.

## **Zusammenfassung**

# Inhaltsverzeichnis

<b>1</b>	<b>Projektbeschreibung</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Ziel der Arbeit . . . . .	1
1.3	Kapitelübersicht . . . . .	1
<b>2</b>	<b>Grundlagen</b>	<b>2</b>
2.1	Schwarmintelligente Superorganismen . . . . .	2
2.1.1	Begriffsdefintion . . . . .	2
2.1.2	Ameisen . . . . .	3
2.1.3	Bienen . . . . .	3
2.2	Agentenbasierte Modellierung . . . . .	4
2.2.1	Agenten . . . . .	4
2.2.2	Agenten-Beziehungen . . . . .	4
2.2.3	Agenten-Umgebung . . . . .	4
<b>3</b>	<b>Schwarmintelligente Algorithmen</b>	<b>5</b>
3.1	Particle Swarm Optimization . . . . .	5
3.1.1	Suchraum . . . . .	5
3.1.2	Partikel . . . . .	5
3.1.3	Algorithmus . . . . .	6
3.2	Ant Colony Optimization . . . . .	7
3.2.1	Suchraum . . . . .	7
3.2.2	Modellierte Ameise . . . . .	7
3.2.3	Algorithmus . . . . .	8
3.3	Bee Colony Optimization . . . . .	9
3.3.1	Suchraum . . . . .	9
3.3.2	Modellierte Biene . . . . .	9
3.3.3	Algorithmus . . . . .	10
<b>4</b>	<b>Modelle und Frameworks</b>	<b>11</b>
4.1	Modelle . . . . .	11
4.1.1	Finden des globalen Minimums einer Funktion mit PSO . . . . .	11
4.1.2	Futtersuche mit ACO & BCO . . . . .	11
4.2	Framework . . . . .	11
4.2.1	Anforderungen . . . . .	11
4.2.2	Übersicht . . . . .	11
4.2.3	Auswahl . . . . .	12
<b>5</b>	<b>Implementierung</b>	<b>12</b>
<b>6</b>	<b>Zusammenfassung</b>	<b>12</b>
<b>7</b>	<b>Anhang 1</b>	<b>15</b>

## Abbildungsverzeichnis

1	Entstehungsbedingungen und Definitionen kollektiver Intelligenz bei Tierschwärmen . . . . .	2
2	PSO, Berechnung von $x_i(t + 1)$ und $v_i(t + 1)$ . . . . .	6

## **Tabellenverzeichnis**

## **Abkürzungsverzeichnis**

**ACO** Ameisenkolonieoptimierung

**BCO** Bienenkolonieoptimierung

**PSO** Partikelschwarmoptimierung



# **1 Projektbeschreibung**

## **1.1 Motivation**

## **1.2 Ziel der Arbeit**

## **1.3 Kapitelübersicht**

## 2 Grundlagen

### 2.1 Schwarmintelligente Superorganismen

Durch die sensorische Verbindung der Tiere wird die Futtersuche und das Abwehren von Gefahren ohne eine zentrale Lenkung bzw. ohne hierarchische Befehlskette bewältigt. Instinktiv verankerte Regeln sorgen dafür, dass auf bestimmte Aktionen der Tiere in vollkommen deterministischer Weise eine Reaktion erfolgt. Aus dieser dezentralen Interaktion entstehen, bei einem Kollektiv von Tieren, intelligente Resultate auf Makroebene.

#### 2.1.1 Begriffsdefinition

Bereits 1911 bezeichnet W. M. Wheeler, amerikanischer Ethologe mit Spezialisierung auf dem Gebiet der Erforschung sozialer Insekten, Kolonien wie die der Bienen und Ameisen als Superorganismen mit emergenten Fähigkeiten<sup>1</sup>. Abbildung 1 zeigt die Definition und Entstehungsbedingungen kollektiver Intelligenz bei Tierschwärmen:



Abbildung 1: Entstehungsbedingungen und Definitionen kollektiver Intelligenz bei Tierschwärmen<sup>3</sup>

- **Interaktion:** Auf Aktion und Reaktion basierende Interaktion gilt als Grundlage für die Definition des Begriffs Schwarmintelligenz. Anlass dafür ist der in den Tieren vorhandene Überlebensinstinkt, der diese veranlasst, sich bewusst an der Gruppe zu beteiligen.
- **Unmittelbares Ergebnis:** Einzelne Tiere führen Handlungen aus ohne Wissen um das Schwarmergebnis. Das Resultat entsteht unmittelbar aus der Handlung des Schwarms und bedarf keiner externen Aggregation und Auswertung.

<sup>1</sup>W.M. Wheeler (1911)

<sup>3</sup>A. Aulinger (2013)

- **Taktische Verbundenheit:** Sowohl die sensuale Verbindung der Tiere als auch der in den Tieren verankerte Instinkt zeugen von der taktischen Verbundenheit des Schwarms bestehend aus festen Aktions- und Reaktionsmustern.

Zusammenfassend beschreibt der Begriff Schwarmintelligenz ein Phänomen aus dem Tierreich zur Selbstorganisation eines Schwarms, um lebensnotwendige Aufgaben gemeinsam auf intelligente Weise zu bewältigen. Dabei vollbringen die Tiere im Schwarm Leistungen, die das Vermögen jedes Einzeltiers übersteigen.

### 2.1.2 Ameisen

Ein Ameisenstaat passt sich ständig an neue Gegebenheiten seiner Umwelt an. Ameisen bilden Staaten mit einigen hundert bis zu mehreren Millionen Individuen. Trotz dieser riesigen Anzahl funktioniert ein Ameisenstaat, da er sich selbst organisiert ohne eine hierarchische Instanz, die einen Überblick über alle Aufgaben besitzt oder diese steuert und verteilt. Stattdessen führen die Handlungen einzelner Ameisen im Zusammenspiel zu einem organisierten Staat, der für die Ameisen sorgt und Nahrung, Brutpflege und Schutz bietet. Für die Informatik besonders interessant sind die Ameisen aufgrund ihrer Fähigkeit effiziente Wege zwischen Futterquellen und dem Ameisenbau ausfindig zu machen<sup>4</sup>.

Ist der Futtervorrat des Ameisenbaus erschöpft verlassen mehrere Ameisen den Ameisenbau gleichzeitig und begeben sich auf die Futtersuche. Sobald eine Ameise eine Futterquelle gefunden hat, nimmt sie eine Gewichtseinheit des Futters mit und begibt sich auf den Rückweg zum Ameisenbau. Dabei setzt die Ameise Pheromone frei die mit der Zeit verfliegen, um den Weg zur Futterquelle zu markieren. Die Ameise die den kürzesten Weg zu einer Futterquelle gefunden hat legt die Strecke zwischen Ameisenbau und Futterquelle häufiger zurück. Die Pheromonspur wird durch das häufige Zurücklegen der Strecke intensiviert und dient als sicherer Wegweiser zur Futterquelle. Mitglieder der Kolonie folgen den intensivsten Pheromonspuren. Ist die Futterquelle erschöpft, löst sich die Pheromonspur auf<sup>5</sup>.

### 2.1.3 Bienen

Um den Gesamtenergiebedarf des Schwarms zu ermitteln, orientiert sich die einzelne Sammlerin an der Wartezeit bei der Übergabe des gesammelten Nektars an die Bienen die den Nektar speichern: Je voller die Futterspeicher, desto länger müssen die Speicherbienen nach leeren Zellen suchen. Je leerer die Speicher, desto schneller wird die Abgabe des Nektars abgewickelt.

Im kilometerweiten Gelände besitzt keine Biene den gesamten geographischen Überblick. Sie entscheidet nur lokal über die Rentabilität der Futterquelle. Kundschafterinnen und Sammlerinnen, die von der Futtersuche wiederkehren, führen einen Tanz auf und zeigen unbeschäftigten Bienen damit die Richtung und Entfernung zur gefundenen Futterquelle. Unbeschäftigte Bienen sehen sich die Tänze der im Bienenstock eintreffenden Bienen an und entscheiden sich anschließend für ihr nächstes Ziel. Sobald eine Futterquelle erschöpft ist brauchen die Sammlerinnen länger beim Sammeln und veranlassen aufgrund der geringeren Anzahl an Tänzen weniger Bienen dazu am gleichen Ort zu sammeln.

---

<sup>4</sup>L. Pintscher (2008)

<sup>5</sup>R. Wehner (2001)

## 2.2 Agentenbasierte Modellierung

Die Agentenbasierte Modellierung erlaubt eine natürliche Beschreibung von Systemen als eine Sammlung autonomer entscheidungsfähiger Agenten in einer gemeinsamen Umgebung, um emergente Phänomene zu analysieren. Das Ziel der Agentenbasierten Modellierung besteht darin, durch die Simulation einer Vielzahl an Agenten, das resultierende Systemverhalten zu untersuchen. Nach C. Macal und M. North<sup>6</sup> verfügt ein agentenbasiertes Modell über folgende Komponenten:

- **Agenten:** Agenten handeln autonom, proaktiv und reaktionär auf der Basis von festgelegten Regeln. Jeder Agent besitzt nur eine eingeschränkte Sicht auf das Gesamtsystem. Sein Wissen über den globalen Zustand des Systems ist immer unvollständig.
- **Agenten-Beziehungen:** Beziehungen zwischen Agenten entstehen durch die proaktive Aktion eines Agenten und die darauffolgende Reaktion eines anderen Agenten oder Interaktionen der Agenten mit der Umgebung.
- **Agenten-Umgebung:** Agenten interagieren innerhalb einer Umgebung mit anderen Agenten und ihrer Umgebung.

### 2.2.1 Agenten

M. Wooldridge und N. Jennings<sup>7</sup> weisen Agenten die Charaktereigenschaften Autonomie, Proaktivität, Reaktivität und die Fähigkeit zur Interaktion durch Kommunikation zu. Die Handlungsautonomie eines Agenten beschränkt sich auf die Fähigkeit, eigenständig zu entscheiden, welche der ihm zur Verfügung stehenden Aktionen situationsbedingt auszuführen ist. Zielvorgaben bestimmen wann ein Agent welche Aktionen ausführt. Proaktivität beschreibt dabei das zielgerichtete Verhalten eines Agenten der selbst aktiv wird, statt nur auf die Umgebung zu reagieren.

### 2.2.2 Agenten-Beziehungen

Zwischen den Systemelementen bestehen Interaktionsbeziehungen. Die technischen Voraussetzungen für die Kommunikation werden von der Umgebung bereitgestellt. Beziehungen lassen sich einteilen in Interaktion zwischen Agenten und Umgebung, Interaktion zwischen Agenten und organisatorisch bedingte Beziehungen. Die Kommunikation zwischen Agenten kann dabei direkt durch Austausch von Nachrichten oder indirekt über die Veränderung der Umgebung erfolgen.

### 2.2.3 Agenten-Umgebung

Nach S. Russell und P. Norvig<sup>8</sup> muss eine geeignete Agenten-Umgebung zugänglich, deterministisch, dynamisch, kontrollierbar und teleologisch sein. Eine zugängliche Umgebung erlaubt den Agenten Zugriff auf ihren Zustand. In der Regel beschränkt sich dieser Zugriff jedoch auf den lokalen Wahrnehmungsbereich eines Agenten und erlaubt somit nur Zugriff auf einen Ausschnitt der Umgebung. Deterministisch ist die Umgebung, sobald der Folgezustand vollständig durch den aktuellen Umgebungszustand und die aktuelle Aktion des Agenten bestimmt ist. Kann sich der Zustand der Umgebung durch Aktionen der Agenten ändern, so handelt es sich um eine dynamische und kontrollierbare Umgebung.

---

<sup>6</sup>C. Macal, M. North (2010)

<sup>7</sup>M. Wooldridge, N. Jennings (1995)

<sup>8</sup>S. Russel, P. Norvig (2009)

## 3 Schwarmintelligente Algorithmen

### 3.1 Particle Swarm Optimization

Die klassische Partikelschwarmoptimierung (PSO) wurde erstmals im Jahr 1995 von J. Kennedy und R. Ebert beschrieben. Die PSO stellt ein, auf der Abfolge von abstrakten Schritten basiertes, Verfahren zur näherungsweisen Lösung von Optimierungsproblemen dar. Zur Lösung des Optimierungsproblems wird eine Population von Partikeln, so lange durch einen Suchraum bewegt, bis eine hinreichende Lösung in einer bestimmten Zeit gefunden wird. Dabei stellt die Position eines Partikels eine potentielle Lösung dar und wird in jedem Zeitschritt neu berechnet<sup>9</sup>. Um die PSO auf ein mathematisches Problem anwenden zu können, müssen zunächst der Suchraum und die Partikel definiert werden<sup>10</sup>.

#### 3.1.1 Suchraum

Der Suchraum  $S \subset \mathbb{R}^n$  ist ein n-dimensionaler Raum, in dem sich die gesamte Population  $X = \{x_1, x_2, \dots, x_n\}$  der Partikel bewegt. Damit eine Lösung gefunden werden kann beschränkt sich der Suchraum auf eine endliche Anzahl an möglichen Positionen, die von den Partikeln besucht werden können.

#### 3.1.2 Partikel

Jedes Partikel  $x_i$  besitzt folgende Eigenschaften im Suchraum  $S$  zum Zeitpunkt  $t$ :

- Position  $x_i(t)$  im Suchraum  $S$  mit einem Funktionswert  $f(x_i(t))$ : Die aktuelle Position des Partikels  $x_i$  zum Zeitpunkt  $t$  repräsentiert eine potentielle Lösung des Optimierungsproblems. Der dazugehörige Funktionswert  $f(x_i(t))$  beschreibt die Güte der potentiellen Lösung.
- Position  $x_{b,i}(t)$  im Suchraum  $S$  mit einem Funktionswert  $f(x_{b,i}(t))$ : Die Position  $x_{b,i}$  ist die bisher beste Position des Partikels  $x_i$  zum Zeitpunkt  $t$ .
- Position  $x_g(t)$  im Suchraum  $S$  mit einem Funktionswert  $f(x_g(t))$ : Die Position  $x_g$  ist die bisher beste Position des Partikelschwarms zum Zeitpunkt  $t$ .
- Geschwindigkeit  $v_i(t)$ : Die Geschwindigkeit  $v_i$  mit der sich der Partikel durch den Suchraum  $S$  zum Zeitpunkt  $t$  bewegt.
- Trägheitskoeffizient  $w$ : Der Trägheitskoeffizient  $w$  bestimmt die Gewichtung der Geschwindigkeit  $v_i(t)$  zum Zeitpunkt  $t + 1$ .
- lokaler Vertrauenskoeffizient  $a_l$ : Der lokale Vertrauenskoeffizient  $a_l$  bestimmt die Gewichtung der lokalen Attraktion, bei der Berechnung der Geschwindigkeit  $v_i(t + 1)$ .
- globaler Vertrauenskoeffizient  $a_g$ : Der globale Vertrauenskoeffizient  $a_g$  bestimmt die Gewichtung der globalen Attraktion, bei der Berechnung der Geschwindigkeit  $v_i(t + 1)$ .

Der Trägheitskoeffizient  $w$  und die Vertrauenskoeffizienten  $a_l$  und  $a_g$  geben an, wie sehr die Partikel sich selbst, ihren eigenen Erfahrungen und den Erfahrungen ihrer Nachbarn vertrauen.

---

<sup>9</sup>Y. Liu (2014)

<sup>10</sup>E. Konstantinos, N. Michael (2010)

### 3.1.3 Algorithmus

```

Initialisiere Partikelschwarm  $X$  im Suchraum  $S$ ;
while Maximale Iterationen nicht erreicht do
    Bestimme für jeden Partikel Position  $x_i(t)$ ;
    Bestimme für jeden Partikel Geschwindigkeit  $v_i(t)$ ;
    Berechne für jeden Partikel den Funktionswert  $f(x_i(t))$ ;
    if  $f(x_i(t))$  besser als  $f(x_{b,i}(t))$  then Setze  $x_{b,i}(t) = x_i(t)$  ;
    if  $f(x_i(t))$  besser als  $f(x_g(t))$  then Setze  $x_g(t) = x_i(t)$  ;
    Berechne neue Geschwindigkeit  $v_i(t + 1)$  des Partikels  $x_i$ ;
    Berechne neue Position  $x_i(t + 1)$  des Partikels  $x_i$ ;
end

```

**Algorithm 1:** PSO Algorithmus

Der abgebildete Algorithmus zeigt den Ablauf der klassischen PSO anhand von Pseudocode. Im ersten Schritt wird die Population  $X$  mit zufälligen Positionen  $x_i$  im Suchraum  $S$  initialisiert. Bis die Anzahl an maximalen Iterationen nicht erreicht ist wird für jeden Partikel geprüft ob der Funktionswert  $f(x_i(t))$  für die Position  $x_i(t)$  besser ist als der Funktionswert  $f(x_i(t - 1))$  im vorherigen Schritt an der Position  $x_i(t - 1)$ . Ist der Funktionswert  $f(x_i(t))$  für die Position  $x_i(t)$  des Partikels besser, wird die bisher beste Position des Partikels  $x_{b,i}(t)$  zu  $x_i(t)$ . Zudem wird für jeden Partikel  $x_i$  geprüft, ob der aktuelle Funktionswert  $f(x_i(t))$  des Partikels besser ist als der globale Funktionswert  $f(x_g(t))$ . Ist dies der Fall so wird die globale beste Position  $x_g(t)$  zu  $x_i(t)$ . Aus diesem Grund wurden in Weiterentwicklungen des Algorithmus Nachbarschaftsbeziehungen entwickelt. Es gibt viele Möglichkeiten die Nachbarschaftsbeziehungen zu realisieren. Im Rahmen dieser Studienarbeit wird darauf verzichtet. Für die nächste Iteration wird die Geschwindigkeit  $v_i(t + 1)$  und die neue Position  $x_i(t + 1)$  des Partikels nach folgenden Formeln berechnet:

$$v_i(t + 1) = wv_i(t) + a_l r_1 (x_{b,i}(t) - x_i(t)) + a_g r_2 (x_g(t) - x_i(t)) \quad (1)$$

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \quad (2)$$

Bei der Berechnung der neuen Geschwindigkeit  $v_i(t + 1)$  wird die Entfernung der Position  $x_g(t)$  zur Position  $x_{b,i}(t)$  berechnet. Dabei fallen die Zufallszahlen  $r_1, r_2 \subset \mathbb{R}$  ins Gewicht, sowie die Konstanten  $w, a_l, a_g$ , mit denen das Verhalten des Schwarms verändert wird. Gilt  $a_l > a_g$ , dann streben die Partikel in Richtung von  $x_{b,i}(t)$ . Ist  $a_g > a_l$ , dann konvergieren die Partikel zur Position  $x_g(t)$ . Abbildung 2 zeigt in grafischer Darstellung die Berechnung der neuen Geschwindigkeit  $v_i(t + 1)$  und der neuen Position  $x_i(t + 1)$ .

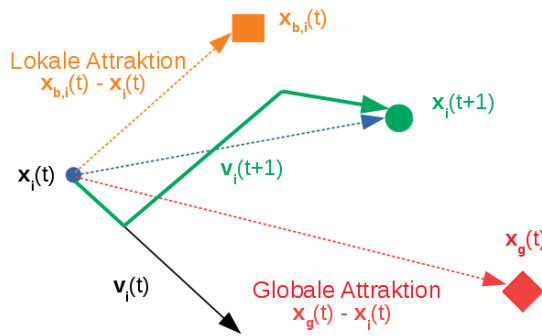


Abbildung 2: Berechnung von  $x_i(t + 1)$  und  $v_i(t + 1)$

## 3.2 Ant Colony Optimization

Die im Jahr 1991 von M. Dorigo vorgestellte Ameisenkolonieoptimierung (ACO) dient, wie die Partikelschwarmoptimierung, zur Lösung von Optimierungsproblemen. Zur näherungsweise Lösung des Optimierungsproblems wird bei der ACO eine Population an modellierten Ameisen instanziiert und durch den vom Optimierungsproblem definierten Suchraum bewegt. Die Positionen der modellierten Ameisen repräsentieren potentielle Lösungen des Optimierungsproblems und werden in jedem Zeitschritt neu berechnet. Darüber hinaus ist jede Position im Suchraum mit einem Pheromonniveau markiert. Je höher das jeweilige Pheromonniveau, desto höher ist die Wahrscheinlichkeit, das sich die modellierten Ameisen in Richtung der mit Pheromonen markierten Positionen bewegen.

### 3.2.1 Suchraum

Der Suchraum  $S \subset \mathbb{R}^n$  ist ein  $n$ -dimensionaler Raum, in dem sich die gesamte Population  $X = \{x_1, x_2, \dots, x_n\}$  der modellierten Ameisen bewegt. Damit eine Lösung gefunden werden kann beschränkt sich der Suchraum auf eine endliche Anzahl an möglichen Positionen  $P = \{p_1, p_2, \dots, p_n\}$  die von den modellierten Ameisen besucht werden können. Jeder Position  $P$  im Suchraum  $S$  ist ein Wert  $\tau_i$  zugeordnet, der das Pheromonniveau der Position beschreibt.

### 3.2.2 Modellierte Ameise

Jede modellierte Ameise besitzt folgende Eigenschaften im Suchraum  $S$  zum Zeitpunkt  $t$ :

- Position  $x_i(t)$  im Suchraum  $S$  mit einem Funktionswert  $f(x_i(t))$ : Die aktuelle Position der modellierten Ameise  $x_i$  zum Zeitpunkt  $t$  repräsentiert eine potentielle Lösung des Optimierungsproblems. Der dazugehörige Funktionswert  $f(x_i(t))$  beschreibt die Güte der potentiellen Lösung.
- Position  $x_g(t)$  im Suchraum  $S$  mit einem Funktionswert  $f(x_g(t))$ : Die Position  $x_g$  ist die bisher beste Position der Ameisenkolonie zum Zeitpunkt  $t$ .
- Teillösungen  $C = \{c_1(t), c_2(t), \dots, c_n(t)\}$  die der modellierten Ameise  $x_i$  zum Zeitpunkt  $t$  zur Verfügung stehen: Die modellierte Ameise  $x_i$  wählt aus der ihr zur Verfügung stehenden Teillösungen  $C$  zum Zeitpunkt  $t$  eine Teillösung. Die gewählte Teillösung entspricht der nächsten Position der modellierten Ameise  $x_i$  zum Zeitpunkt  $t + 1$ .
- Wahrscheinlichkeiten  $W = \{w_{c_1}(t), w_{c_2}(t), \dots, w_{c_n}(t)\}$  zur Wahl von  $x_i(t+1)$ : Die Wahrscheinlichkeiten  $W$  werden für die, der modellierten Ameise zur Verfügung stehenden, Teillösungen  $C$  berechnet. Die Teillösung mit der höchsten Wahrscheinlichkeit wird von der modellierten Ameise ausgewählt.
- Parameter  $\alpha$ :
- Parameter  $\beta$ :

### 3.2.3 Algorithmus

```

Initialisiere Ameisenkolonie  $X$  im Suchraum  $S$ ;
while Maximale Iterationen nicht erreicht do
    Berechne für jede modellierte Ameise  $x_i$  die Wahrscheinlichkeiten  $W$  für die
        möglichen Teillösungen  $C$  ausgehend von Position  $x_i(t)$ ;
    Bestimme für jede modellierte Ameise neue Position  $x_i(t+1)$ ;
    if  $f(x_i(t+1))$  besser als  $f(x_g(t))$  then Setze  $x_g(t) = x_i(t+1)$ ;
    Berechne für jede gewählte Teillösung die neue Pheromonmarkierung  $\tau_i(t+1)$ 
        für Position  $P$  im Suchraum  $S$ ;
end

```

**Algorithm 2:** ACO Algorithmus

Im ersten Schritt des abgebildeten Algorithmus wird die Population  $X$  an einer bestimmten Position  $P$  im Suchraum  $S$  initialisiert. Solange die Anzahl an maximalen Iterationen nicht erreicht ist berechnet jede Ameise die Wahrscheinlichkeiten  $W$  für die ihr zur Verfügung stehenden Teillösungen  $C$  mit der Formel:

$$W(c_i|x_i(t)) = \frac{\tau_{c_i}^\alpha(t) \cdot [f(c_i(t))]^\beta}{\sum_{c \in C} \tau_c(t)^\alpha \cdot [f(c(t))]^\beta} \quad (1)$$

Die Wahrscheinlichkeit ist abhängig von dem Pheromonniveau  $\tau_{c_i}$  der jeweiligen Teillösung und der Güte der Teillösung, bestimmt durch  $f(c_i(t))$ . Die Parameter  $\alpha$  und  $\beta$  verändern das Verhalten der modellierten Ameisen. Gilt  $\alpha > \beta$  dann ist das Pheromonniveau der Teillösung ausschlaggebender als die Güte der Teillösung für die Berechnung der Wahrscheinlichkeiten  $W$ . Sind die Wahrscheinlichkeiten  $W$  für alle Teillösungen  $C$  berechnet, entscheidet sich die modellierte Ameise für die Teillösung mit der höchsten Wahrscheinlichkeit. Die von der modellierten Ameise gewählte Teillösung entspricht der neuen Position  $x_i(t+1)$ . Im Anschluss wird für jede modellierte Ameise  $x_i(t)$  geprüft, ob der aktuelle Funktionswert  $f(x_i(t))$  besser ist als der globale Funktionswert  $f(x_g(t))$ . Ist dies der Fall so wird die global beste Position  $x_g(t)$  zu  $x_i(t)$ . Am Ende der Iteration werden die Pheromonmarkierungen mit folgender Formel aktualisiert:

$$\tau_i(t+1) = (1 - \rho)\tau_i(t) + \sum_{x_i \in X} \Delta\tau_i(t)^C \quad (2)$$

Hierbei ist  $\rho$  der konstante Verdunstungsfaktor der Pheromonmarkierungen. Die an der gewählten Teillösung hinterlassene Pheromonmenge  $\Delta\tau_i(t)^C$  berechnet sich durch die Formel:

$$\Delta\tau_i(t)^C = f(c_i(t)) \quad (3)$$



### 3.3 Bee Colony Optimization

Die im Zeitraum 1999 bis 2003 von D. Teodorović und P. Lučić entwickelte Bienenkolonieoptimierung (BCO) stellt, wie die PSO und ACO, ein metaheuristisches Verfahren zur Lösung von Optimierungsproblemen dar. Die zur näherungsweisen Lösung des Optimierungsproblems, durch den Suchraum bewegten modellierten Bienen repräsentieren potentielle Lösungen des Optimierungsproblems, über ihre Position im Suchraum.

#### 3.3.1 Suchraum

Der Suchraum  $S$  der BCO unterscheidet sich nicht von dem in Kapitel 3.1.1 beschriebenen Suchraum der PSO.

#### 3.3.2 Modellierte Biene

Jede modellierte Biene besitzt folgende Eigenschaften im Suchraum  $S$  zum Zeitpunkt  $t$ :

- Position  $x_i(t)$  im Suchraum  $S$  mit einem Funktionswert  $f(x_i(t))$ : Die aktuelle Position der modellierten Biene  $x_i$  zum Zeitpunkt  $t$  repräsentiert eine potentielle Lösung des Optimierungsproblems. Der dazugehörige Funktionswert  $f(x_i(t))$  beschreibt die Güte der potentiellen Lösung.
- Position  $x_g(t)$  im Suchraum  $S$  mit einem Funktionswert  $f(x_g(t))$ : Die Position  $x_g$  ist die bisher beste Position der Bienenkolonie zum Zeitpunkt  $t$ .
- Loyalitätswahrscheinlichkeit  $l_i(t)$ : Die Loyalitätswahrscheinlichkeit  $l_i(t)$  der modellierten Biene  $x_i$  bestimmt den Status  $Z$  zum Zeitpunkt  $t$ .
- Status  $Z$ : Ist die Loyalitätswahrscheinlichkeit  $l_i(t)$  größer als eine Zufallszahl so wird die modellierte Biene  $x_i$  zum Zeitpunkt  $t$  in den Status Loyal versetzt. Die modellierte Biene erhält den Status Frei sobald die Loyalitätswahrscheinlichkeit kleiner ist als die Zufallszahl.
- (optional) Rekrutierungsbienen  $R = \{r_1(t), r_2(t), \dots, r_n(t)\}$  die der modellierten Biene  $x_i$  zum Zeitpunkt  $t$  zur Verfügung stehen. Die modellierte Biene  $x_i$  wählt aus der ihr zur Verfügung stehenden Rekrutierungsbienen  $R$  zum Zeitpunkt  $t$  eine Rekrutierungsbiene.
- (optional) Richtung  $v_i(t)$ : Die Richtung  $v_i(t)$  von der aktuellen Position  $x_i(t)$  der modellierten Biene zur gewählten Rekrutierungsbiene  $r_i(t)$ .

Befindet sich eine modellierte Biene  $x_i$  im Status Loyal besitzt sie zum Zeitpunkt  $t$  keine Rekrutierungsbienen  $R$  sondern ist ihrer Lösung  $x_i(t)$  gegenüber loyal und wird diese nicht verändern. Ebenso besitzt sie daher keine Richtung  $v_i(t)$ .

### 3.3.3 Algorithmus

```

Initialisiere Bienenkolonie  $X$  im Suchraum  $S$ ;
while Maximale Iterationen nicht erreicht do
    Bestimme für jede modellierte Biene Position  $x_i(t)$  und berechne den
    Funktionswert  $f(x_i(t))$ ;
    Bestimme  $n$  Rekrutierungsbiene  $R$  mit den besten Funktionswerten  $f(x_i(t))$ ;
    Berechne für jede modellierte Biene die Loyalitätswahrscheinlichkeit  $l_i(t)$ ;
    if  $l_i(t)$  größer als Zufallszahl then
        | Versetze Biene in den Status Loyal;
    else
        | Versetze Biene in den Status Frei;
    end
    if Biene im Zustand Frei then
        | Berechne Rekrutierungswahrscheinlichkeiten  $W$  für Rekrutierungsbiene  $R$ 
        | und wähle Rekrutierungsbiene  $r_i(t)$ ;
        | Bestimme Richtung  $v_i(t)$  von Position  $x_i(t)$  zu  $r_i(t)$ ;
        | Setze neue Position  $x_i(t+1) = x_i(t) + v_i(t)$ ;
    else
        | Setze neue Position  $x_i(t+1) = x_i(t)$ ;
    end
    if  $f(x_i(t+1))$  besser als  $f(x_g(t))$  then Setze  $x_g(t) = x_i(t+1)$ ;
end

```

**Algorithm 3:** BCO Algorithmus

$$l_i(t) = e^{-\frac{f(x_g(t)) - f(x_i(t))}{u}} \quad (1)$$

$$r_i(t) = \frac{f(r_i(t))}{\sum_{r \in R} f(r(t))} \quad (2)$$

## 4 Modelle und Frameworks

### 4.1 Modelle

#### 4.1.1 Finden des globalen Minimums einer Funktion mit PSO

#### 4.1.2 Futtersuche mit ACO & BCO

### 4.2 Framework

#### 4.2.1 Anforderungen

Die Anforderungen an das Framework zur Erstellung der in Kapitel 4.1 vorgestellten Modelle und anschließender grafischer Simulation sind:

- **Open-Source + kostenlos:** Der Quelltext des Frameworks ist öffentlich. Zudem kann das Framework kostenlos genutzt werden.
- **Aktualität:** Es ist eine aktuelle und stabile Version des Frameworks verfügbar.
- **Einsetzbarkeit:** Das Framework kann auf dem Betriebssystem Linux verwendet werden und unterstützt die Programmiersprache Java.
- **Visualisierung:** Modelle werden grafisch in 2D simuliert.
- **Dokumentation + Tutorial:** Das Framework ist umfangreich dokumentiert und besitzt ein Tutorial für Einsteiger.
- **Ausführungsgeschwindigkeit der Simulation:** Die Ausführungsgeschwindigkeit der Simulation ist hoch trotz rechenintensiven Modellen mit vielen Agenten und Iterationen.

#### 4.2.2 Übersicht

**NetLogo** NetLogo ist eine Open-Source-Plattform und wurde 1999 von U. Wilensky entwickelt. Die neueste Version 6.0.3 wurde im März 2018 veröffentlicht und steht für die Betriebssysteme Mac, Windows und Linux zur Verfügung. Die proprietäre funktionale Programmiersprache und die ausführliche Dokumentation machen es Einsteigern ohne Programmierkenntnisse einfach erste eigene Modelle zu implementieren. Das Fehlen objektorientierter Features schränkt die Funktionalität und Erweiterbarkeit des Frameworks jedoch ein. Darüber hinaus verfügt NetLogo über keine explizite Unterstützung für die Trennung zwischen Agenten, Agenten-Beziehungen und Agenten-Umgebung.

**Mason** Mason ist eine Java-Bibliothek, die ein breitgefächertes Repertoire an Funktionen für die agentenbasierte Modellierung bereitstellt. Die aktuelle Version 1.9 wurde im Juni 2015 veröffentlicht. Eine umfangreiche Dokumentation der Open-Source-Bibliothek und ein einsteigerfreundliches Tutorial erlauben es innerhalb weniger Stunden eine erste lauffähige Simulation zu erstellen. Kombiniert mit der JFreeChart-Bibliothek ist es zudem möglich die Simulationen in grafischer Form auszuwerten.

**Swarm** Swarm ist eine der ersten ABM-Plattformen. Ursprünglich wurde Swarm 1997 am Santa Fe Institute entwickelt und veröffentlicht. Die ABM-Plattform Swarm war zuerst nur auf UNIX-basierten Betriebssystemen einsetzbar und unterstützte nur die Programmiersprache Objective-C. Heute steht Swarm für die Betriebssysteme Mac, Windows und Linux zur Verfügung und unterstützt neben Objective-C auch Java. Das Framework ist Open-Source. Version 2.4.1 wurde im April 2009 veröffentlicht.

**Repast** Repast ist eine Open-Source-Plattform für die agentenbasierte Modellierung und Simulation. Die Software ist für die Betriebssysteme Mac, Windows und Linux verfügbar. Es existieren zwei verschiedene Editionen der Plattform. Repast Symphony implementiert grundlegende und erweiterte Konzepte der agentenbasierten Modellierung. Unterstützt werden die Programmiersprachen Java, Logo und Groovy. Repast High Performance Computing ist ein agentenbasiertes Modellierungssystem speziell entwickelt für die parallele Ausführung der Simulationen auf verteilten Systemen. Zur Modellentwicklung mit dieser Edition von Repast werden umfangreiche Programmierkenntnisse in C++ vorausgesetzt. Die stabile Version 2.5 von Repast Symphonie wurde im Oktober 2017 veröffentlicht.

**Zusammenfassung** S.F. Railsback, S.L. Lytinen und S.K. Jackson untersuchten im Jahr 2006 verschiedene Plattformen für die agentenbasierte Modellierung und Simulation. Ihre Ergebnisse über die oben genannten ABM-Plattformen können folgendermaßen zusammengefasst werden:

- **NetLogo:** Netlogo ist einfach zu bedienen und besitzt eine umfangreiche Dokumentation. NetLogo ist eine gute Wahl für simple Modelle bei denen die Ausführungszeit der Simulation eine untergeordnete Rolle spielt. Außerdem eignet sich NetLogo zur Erstellung eines Modellprototyps. Dieser Prototyp kann anschließend in anderen ABM-Plattformen erweitert werden.
- **Mason:** Mason ist eine gute Wahl für erfahrene Programmierer, die an rechenintensiven Modellen mit vielen Agenten und Iterationen arbeiten. Ein Vorteil von Mason ist die Reproduzierbarkeit der Simulationsergebnisse auf verschiedenen Betriebssystemen.
- **Swarm:** Swarm ist stabil, klein, gut organisiert und unterstützt komplexe Modelle. Voraussetzung für einen schnellen Einstieg ist Programmiererfahrung mit Objective-C.
- **Repast:** Repast ist die vollständigste ABM-Plattform und hat eine vergleichsweise schnelle Ausführungszeit der Simulationen. Die Dokumentation ist teilweise unvollständig.

#### 4.2.3 Auswahl

## 5 Implementierung

## 6 Zusammenfassung

## Literatur

- [1] A. Aulinger. Kollektive Intelligenz: Methoden, Erfahrungen und Perspektiven. Entstehungsjahr: 2009.
- [2] C. Macal, M. North. Tutorial on agent-based modelling and simulation. Journal of Simulation 4, Nr. 1, Seite 151-162. Entstehungsjahr: 2010.
- [3] L. Pintscher. Schwarmintelligenz. Universität Karlsruhe. Entstehungsjahr: ohne Jahr.

Weblink: <http://lydiapintscher.de/uni/schwarmintelligenz.pdf>  
Einsichtnahme: 14.01.2018

- [4] S. Russel, P. Norvig. Artificial Intelligence: A Modern Approach. Upper Saddle River. Entstehungsjahr: 2009.
- [5] W.M. Wheeler. The Ant Colony as an Organism. Journal of Morphology Volume 22, Issue 2, Seite 307-325. Entstehungsjahr: 1911.

Weblink: <http://www3.interscience.wiley.com/journal/109914213/abstract>  
Einsichtnahme: 04.01.2018

- [6] M. Wooldridge, N. Jennings. Intelligent Agents: Theory and Practice. Knowledge Engineering Review 10, Nr. 2, Seite 115-152. Entstehungsjahr: 1995.
- [7] Y. Liu. Partikelschwarmoptimierung für diskrete Probleme. Technische Universität München. Entstehungsjahr: 2014.

Weblink: <http://lydiapintscher.de/uni/schwarmintelligenz.pdf>  
Einsichtnahme: 14.01.2018

## **Verzeichnis der Anhänge**

## **7 Anhang 1**